Lösningsförslag till tentamen

Kurs	**Objektorienterad programmering**
Tentamensdatum	**2013-08-28**
Program	**D2**
Läsår	**2012/2013, lp 1**
Examinator	**Uno Holmer**

Uppgift 1 (10 p)

Ingen lösning ges. Se kurslitteraturen.

Uppgift 2 (10 p)

```
public class CounterButton extends JButton implements ActionListener {
    private int counterValue;
    private Actor actor;

    public CounterButton(int counterValue,Actor actor) {
        this.counterValue = counterValue;
        this.actor = actor;
        setText("" + counterValue);
        addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        --counterValue;
        if ( counterValue == 0 ) {
            setText("--");
            setEnabled(false); // alt. removeActionListener(this);
        } else {
            setText("" + counterValue);
            actor.act();
        }
    }
}
```

Uppgift 3 (10 p)

a) (5 p)

`1-3:` obj har statisk typ Base och dynamisk typ Sub

`1: Sub.f1`	`f1` definieras om i `Sub`
`2: Base.f2++Sub.f2`	`f2` definieras om i `Sub` och den börjar med att anropa `Base.f2` explicit
`3: Base.f3`	`f3` definieras ej om i `Sub` utan ärvs från `Base`

`4-7: obj2` har statisk och dynamisk typ `Sub`. Subklassens metoder anropas utom i fall 6 där `f3` ärvs från basklassen.

```
4: Sub.f1
5: Base.f2++Sub.f2
6: Base.f3
7: Sub.f4
```

b) (5 p)

`Int obj1 = new Int();`
Gränssnitt är abstrakta klasser och sådana får ej instansieras.

`Base obj2 = new Base();` `Base` är abstrakt.

`Sub1 obj3 = new Sub1();` Se nedan.

`Sub2 obj4 = new Sub2();` Korrekt.

Klassen `Sub1` kan ej kompileras eftersom den abstrakta metoden `h` ej är implementerad vare sig i `Base` eller i `Sub1`. Notera att h ej måste definieras i `Base`. Eftersom klassen är abstrakt kan definitionen av `h` skjutas upp till någon subklass längre ner i hierarkin. Alla mellanliggande klasser blir då abstrakta.

Uppgift 4 (2+4+2 p)

a) Om man definierar equals enligt förslaget så blir den inte transitiv. Om avstånden mellan p och q, resp. q och r, är inom toleransen, så ger ju inte det någon information om avståndet mellan p och r, d.v.s. p.equals(q) och q.equals(r) kan båda returnera sant, men p.equals(r) kan returnera sant eller falskt, beroende på de tre punkternas inbördes orientering i planet. Om equals är transitiv måste p.equals(r) returnera sant på ovanstående premisser.

b)

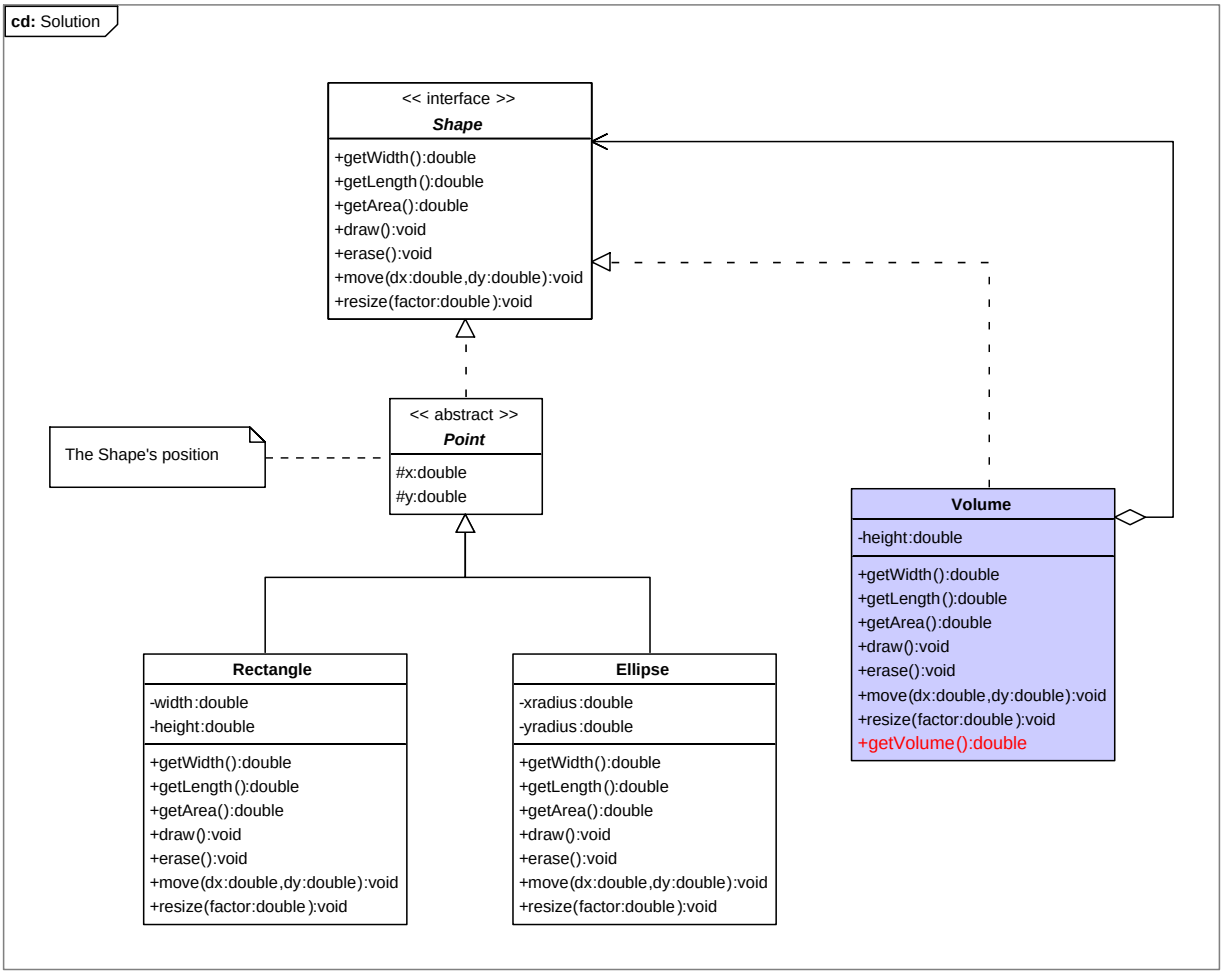
```
public final boolean equals(Object other) {
   if ( this == other )
       return true;
   if ( other instanceof Prenumerant ) {
       Prenumerant o = (Prenumerant)other;
       return (name == null ? o.name == null) :
                   name.equals(o.name)
              &&
              (address == null ? o.address == null) :
                   address.equals(o.address);
   }
   return false;
}
```

c) Observera att `hashCode` bara får baseras på de instansvariabler som används i `equals`. Hashvärdet skall alltså inte bero av telefonnumret.

```
public int hashCode() {
    int code = 123;
    code = 37*code + name.hashCode();
    code = 37*code + address.hashCode();
    return code;
}
```

Uppgift 5 (3+6+1 p)

a) Observera att Volume bör implementera gränssnittet Shape, inte ärva från Point.



b)

```
public class Volume implements Shape
{
    private Shape obj; // The wrapped object
    private double height;

    public Volume(Shape obj,double height)
    {
        this.obj = obj;
        this.height = height;
    }

    // Delegate to the wrapped object
    public double getWidth() { return obj.getWidth(); }
    public double getLength() { return obj.getLength(); }
    public double getArea() { return obj.getArea(); }
    public void draw() { obj.draw(); }
    public void erase() { obj.draw(); }
    public void move(double dx,double dy) { obj.move(dx,dy); }
    public void resize(double factor) {
        height *= factor;
        obj.resize(factor);
    }

    // This is the actual decoration code
    public double getVolume() {
        return obj.getArea()*height;
    }
}
```

c) A cube with side length 10, with a corner positioned in origo:

```
Volume cube = new Volume(new Rectangle(0,0,10,10),10);
System.out.println(cube.getVolume());
```

Uppgift 6 (12 p)

```
public class ResultProcessor
{
    private BufferedReader resultsReader;      // Reads the text file
                                               // containing the exam results

    private Map<String,ExamGrade> examResults; // Maps cid:s to exam results
    private String courseCode;                 // Obtained from the results
                                               // file name
    private ObjectInputStream databaseStream;  // The "LADOK" database

    public ResultProcessor(String databaseFile,String resultsFile)
    {
        try {
            databaseStream =
                new ObjectInputStream(
                    new FileInputStream(databaseFile));

            resultsReader = new BufferedReader(
                                new FileReader(resultsFile));

            courseCode = getNameBase(resultsFile);
            processFiles();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void processFiles() throws IOException {
        examResults = makeResultsTable();
        try {
            StudentRecord student =
                (StudentRecord)databaseStream.readObject();
            while ( student != null ) {
                if ( student.isRegistred(courseCode) &&
                     examResults.containsKey(student.getCid()))
                {
                    ExamGrade eg = examResults.get(student.getCid());
                    MailServer server = MailServer.getInstance();
                    server.sendmail(student.getContact().getEmail(),
                                    "Exam result " + courseCode,
                                    "Hello " + student.getName() +
                                    "!\nYour marks: " + eg.getMarks() +
                                    ", grade: " + eg.getGrade() + "\n");
                }
                student = (StudentRecord)databaseStream.readObject();
            }
        }
        catch (Exception e)  {
            e.printStackTrace();
            return;
        }
    }
...
```

```
    private Map<String,ExamGrade> makeResultsTable() throws IOException {
        Map<String,ExamGrade> table = new HashMap<String,ExamGrade>();

        String line = resultsReader.readLine();
        while ( line != null ) {
            String[] fields = line.split(",");
            if ( fields.length != 3 )
                throw new IOException(”makeResultsTable: malformed input”);
            table.put(fields[0],
                      new ExamGrade(Integer.parseInt(fields[1]),fields[2]));
            line = resultsReader.readLine();
        }

        return table;
    }

    // denna var given
    String getNameBase(String fileName) {
        int i = fileName.indexOf('.');
        if ( i == -1 )
            return fileName;
        else
            return fileName.substring(0,i);
    }
}
```